AI & SUSTAINABILITY IN VET EDUCATION

Name of the Author/s: Sucessos Criativos, Lda | P. Porto

Title of WP: WP3 – AI-SAVES@ProjectHub

Activity Reference: 3.2 – Project template for AI-SAVES@ProjectHub

European Vocational Training Association

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Table of Contents

1. Project Title: 2

2. Project Category: 2

3. Project Description: 2

5. Target Audience: 2

6. Pre-Requisite knowledge: 2

7. Project Objectives: 3

8. Project Timeline: 3

9. Resources Provided: 3

10. Tools and Technologies Used: 4

11. Step-by-Step Implementation Plan: 6

12. Evaluation Criteria: 15

13. Languages Available: 15

14. Contributors: 15

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

1. Project Title:

Real-Time Trash Segmentation System with Raspberry Pi and YOLOv11

---

2. Project Category:

- AI in IoT

---

3. Project Description:

This project develops an advanced computer vision system using Raspberry Pi and YOLOv11 neural network to detect and segment trash objects in real-time through camera feed. The system provides pixel-perfect segmentation masks with visual overlays, making it ideal for waste management applications, environmental monitoring, and recycling automation. Students will learn to implement deep learning models on edge devices while contributing to environmental sustainability through intelligent waste detection.

---

4. Keywords:

EComputer Vision, YOLOv11, Segmentation, Raspberry Pi, Environmental Monitoring, Waste Management, Deep Learning, Real-time Processing, Edge AI

---

5. Target Audience:

Both Educators and Students

---

6. Pre-Requisite knowledge:

- Intermediate Python programming (classes, functions, file handling)
- Basic understanding of computer vision concepts

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

- Familiarity with neural networks and deep learning fundamentals
- Experience with Raspberry Pi GPIO and camera module
- Knowledge of OpenCV library basics
- Understanding of object detection and segmentation concepts
- Basic Linux command line operations

---

7. Project Objectives:

- Introduce students to advanced computer vision and deep learning on edge devices
- Teach real-time object segmentation using state-of-the-art YOLO models
- Develop practical skills in AI model deployment and optimization
- Raise awareness about environmental challenges and AI-driven solutions
- Demonstrate pixel-perfect segmentation techniques for precise object detection
- Encourage development of sustainable technology solutions
- Build expertise in camera-based AI systems and real-time processing

---

8. Project Timeline:

---

9. Resources Provided:

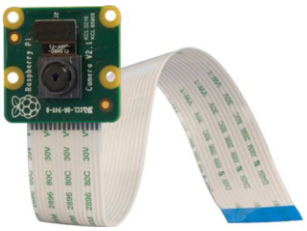
- Raspberry Pi 5
- Raspberry Pi Camera Module v2 or v3
- Keyboard for system control
- Python code templates and libraries
- YOLOv11 training Jupyter notebook
- Sample trash dataset for training (from Kaggle)
- Step-by-step implementation guide
- Troubleshooting documentation

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

10. Tools and Technologies Used:

10.1 Hardware:

MATERIAL	
Raspberry Pi 5	
27W USB Type-C Power Supply for Raspberry Pi 5	
Raspberry Pi Camera Board V2 8MP 1080p (IM8)	
Official FPC camera cable for Raspberry Pi 5 200mm (IM11)	

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

10.2 Software:

- **Operating System:** Raspberry Pi OS
- **Programming Language:** Python 3.8+
- **Development Environment:** Thonny IDE or VS Code
- **Model Training Platform:** Google Colab with GPU acceleration
- **AI Framework:** YOLOv11 (You Only Look Once v11)
- **Model Format:** PyTorch (.pt)

Libraries: time, cv2, YOLO, PiCamera2, os, kagglehub, ultralytics.YOLO, shutil, yaml, pathlib.Path, cv2, numpy, sklearn, glob, matplotlib.pyplot, seaborn, PIL.Image, PIL.ImageDraw, PIL.ImageFont, pandas, json, datetime, torch, tqdm

---

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

11. Step-by-Step Implementation Plan:

11.1 Hardware Setup

Camera Module Installation

1. **Power down** the Raspberry Pi completely
2. **Connect the camera module** to the CSI port
3. **Secure the connection** and ensure proper seating
4. **Power up** and enable camera interface

Hardware Verification

```
# Test camera detection
vcgencmd get_camera
# Expected output: supported=1 detected=1

# Test camera functionality
libcamera-hello --timeout 5000
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

11.2 Model Training (Google Colab)

This section outlines the training process for the YOLOv11-based trash segmentation model using Google Colab with GPU acceleration. The objective is to develop a lightweight model optimized for real-time inference on a Raspberry Pi device.

Environment Setup

1. Open Google Colab.
2. Enable GPU:
   Runtime → Change runtime type → Hardware accelerator: GPU.
3. Upload and open the notebook `train_model.ipynb`.
4. Install required libraries and configure training parameters.

Dataset Configuration

- Dataset sourced from Kaggle, containing annotated images of marine waste.
- Annotations include pixel-level segmentation masks and class labels.
- Converted to YOLOv11 segmentation-compatible format.

Training Process

- **Preprocessing**: Image resizing, normalization (rotation, flipping, brightness adjustments).
- **Model**: YOLOv11n-seg (a lightweight segmentation variant optimized for edge devices).
- **Training execution**
- **Evaluation**: Automatic performance assessment after each epoch using metrics such as mAP and IoU.
- **Export**: Final model saved in `.pt` format for deployment on Raspberry Pi

Output

- The resulting `yolov11_seg_trash.pt` model is optimized for inference on Raspberry Pi.
- Provides real-time object detection with pixel-accurate segmentation masks.
- Output visualizations and metrics can be reviewed directly within Colab.

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

11.3 Software Setup

Now that the hardware is ready, it's time to implement the trash segmentation system.

1. Create Virtual Environment

```
# Create project directory
mkdir trash_segmentation_system
cd trash_segmentation_system

# Create virtual environment
python3 -m venv venv

# Activate virtual environment
source venv/bin/activate
```

2. Install Dependencies

```
# Install requirements
pip install -r requirements.txt
```

3. Add the Code

- Copy the trash segmentation system Python code to `main.py`
- Make sure the file is in your project directory

4. Add the model

- Copy the trash segmentation system model to `model/yolov11_seg_trash.pt`
- Make sure the file is in your project directory

5. Run the System

```
# Make sure virtual environment is activated
source venv/bin/activate

# Run the light trash segmentation system
python main.py
```

6. Stop the System

- Press Ctrl+C to stop the program safely

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

System Behavior

Startup Sequence

- Model loading from `yolov11_seg_trash.pt`
- Camera initialization (800x600 RGB888)
- System displays "✅ Model loaded successfully"
- Real-time detection begins

Operational States

- **Normal Operation**: Continuous frame capture and processing
- **Detection Active**: Colored segmentation masks overlay detected trash
- **No Detection**: System continues monitoring
- **Error Recovery**: Automatic handling of processing failures

Visual Feedback

- **Segmentation Masks**: Colored overlays show exact object boundaries
- **Bounding Boxes**: Rectangle frames around detected objects
- **Labels**: Object type and confidence score display
- **Real-time Display**: OpenCV window shows live results

Interactive Controls

- **Termination**: Ctrl+C for safe shutdown
- **Resource Cleanup**: Automatic camera and window closure
- **Console Output**: Detection counts and status messages

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

CODE

```
import cv2
import numpy as np
from picamera2 import Picamera2
from ultralytics import YOLO
import time
import os

# Initialize the Picamera2
picam2 = Picamera2()
picam2.preview_configuration.main.size = (800, 600)
picam2.preview_configuration.main.format = "RGB888"
picam2.preview_configuration.align()
picam2.configure("preview")
picam2.start()

# Load the custom YOLO segmentation model
print("Loading segmentation model...")
model = YOLO("model/yolov11_seg_trash.pt")
print("✅ Model loaded successfully")

# Configure parameters
conf_threshold = 0.01  # Confidence threshold
colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255), (255, 255, 0), (255, 0, 255),
          (0, 255, 255), (128, 0, 128), (255, 165, 0), (255, 192, 203), (165,
42, 42)]

print("Starting real-time detection...")

try:
    while True:
        # Capture frame-by-frame
        frame = picam2.capture_array()

        # Convert from RGB to BGR for OpenCV (if necessary)
        frame_bgr = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
        img_height, img_width = frame_bgr.shape[:2]

        # Run YOLO inference
        results = model(frame_bgr, conf=conf_threshold, verbose=False)

        # Create copy for overlay
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
        overlay = frame_bgr.copy()

        # Process results following your successful method
        for result in results:
            # Check if there are masks and detections
            if result.masks is not None and len(result.masks) > 0:
                # Get mask data, boxes and confidences
                masks = result.masks.data.cpu().numpy()
                boxes = result.boxes.xyxy.cpu().numpy()
                confidences = result.boxes.conf.cpu().numpy()
                class_indices = result.boxes.cls.cpu().numpy()

                print(f"Detected {len(masks)} masks")

                # Process each prediction
                    for i, (mask, box, conf, cls_idx) in enumerate(zip(masks,
boxes, confidences, class_indices)):
                    if conf > conf_threshold:
                        # Resize mask to image size
                                    mask_resized = cv2.resize(mask, (img_width,
img_height))

                        # Get color for this class
                        color = np.array(colors[int(cls_idx) % len(colors)])

                        # Create colored mask overlay
                        mask_indices = mask_resized > 0.5
                           overlay[mask_indices] = overlay[mask_indices] * 0.6 +
color * 0.4

                        # Draw bounding box
                        x1, y1, x2, y2 = box.astype(int)
                                    cv2.rectangle(overlay, (x1, y1), (x2, y2),
color.tolist(), 2)

                        # Get class name
                                    class_name = result.names[int(cls_idx)] if
hasattr(result, 'names') else f"Class {int(cls_idx)}"

                        # Add label with confidence
                        label = f"{class_name}: {conf:.2f}"
                           (text_width, text_height), _ = cv2.getTextSize(label,
cv2.FONT_HERSHEY_SIMPLEX, 0.5, 2)
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
                        # Background for text
                                cv2.rectangle(overlay, (x1, y1-text_height-10),
(x1+text_width, y1), color.tolist(), -1)
                                    cv2.putText(overlay, label, (x1, y1-5),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)

                        # Optional: draw contours for better definition
                        mask_binary = (mask_resized > 0.5).astype(np.uint8)
                                contours, _ = cv2.findContours(mask_binary,
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
                         cv2.drawContours(overlay, contours, -1, color.tolist(),
2)

            else:
                # If there are detections but no masks (only bounding boxes)
                if result.boxes is not None and len(result.boxes) > 0:
                    boxes = result.boxes.xyxy.cpu().numpy()
                    confidences = result.boxes.conf.cpu().numpy()
                    class_indices = result.boxes.cls.cpu().numpy()

                            for box, conf, cls_idx in zip(boxes, confidences,
class_indices):
                        if conf > conf_threshold:
                            x1, y1, x2, y2 = box.astype(int)
                            color = colors[int(cls_idx) % len(colors)]

                            # Draw only bounding box if no masks available
                               cv2.rectangle(overlay, (x1, y1), (x2, y2), color,
2)

                                    class_name = result.names[int(cls_idx)] if
hasattr(result, 'names') else f"Class {int(cls_idx)}"
                            label = f"{class_name}: {conf:.2f}"
                                      cv2.putText(overlay, label, (x1, y1-10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

        # Display frame with overlays
        cv2.imshow("YOLO Segmentation", overlay)


except KeyboardInterrupt:
    print("\nInterrupted by user")
except Exception as e:
    print(f"Error: {e}")
```

Co-funded by the European Union

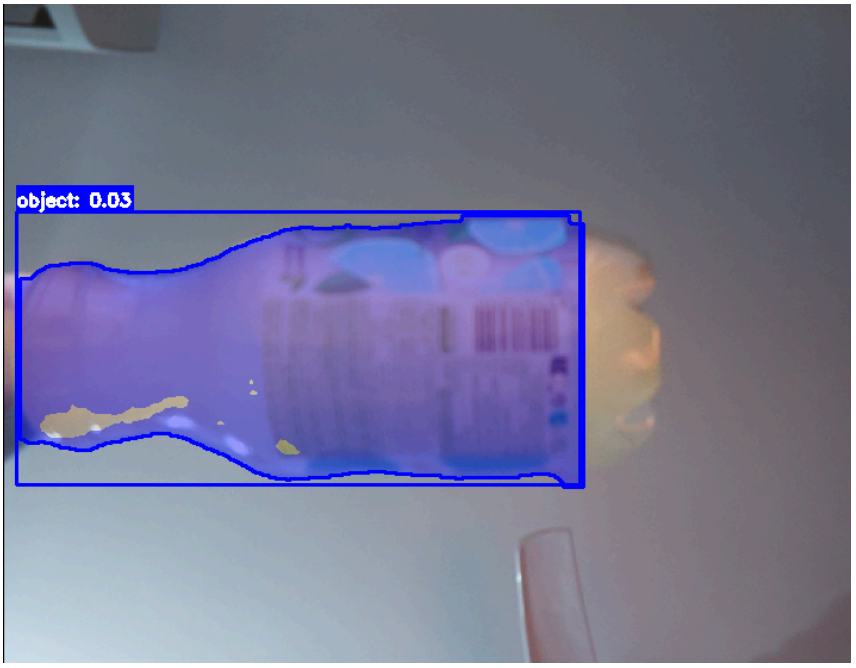
This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

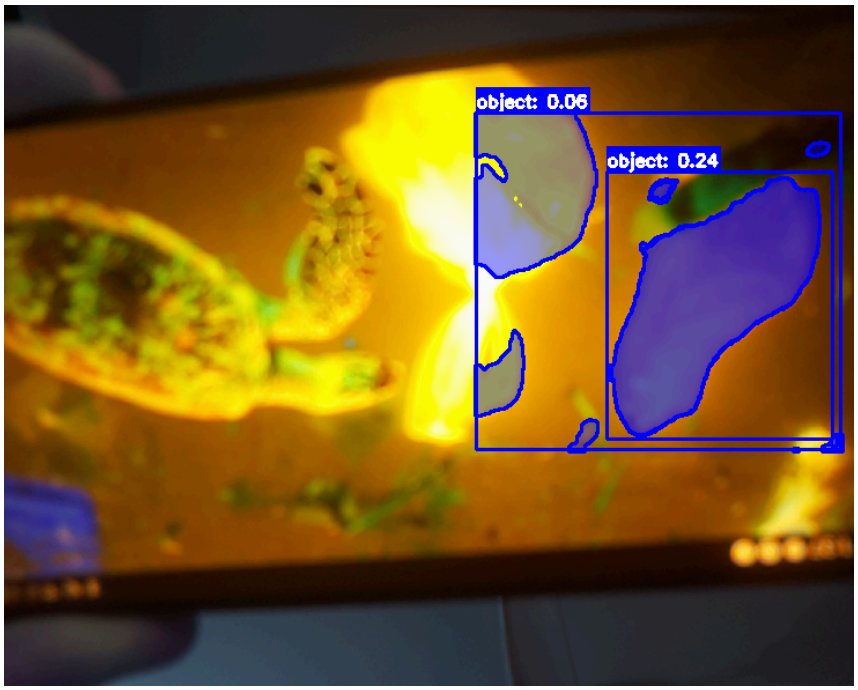
```
finally:
    # Clean up resources
    print("Closing application...")
    picam2.stop()
    cv2.destroyAllWindows()
    print("✅ Resources freed successfully")
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Now, you can run the program and see the result:





Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

12. Evaluation Criteria:

- **Model Training Execution (60%):** Correct execution of the Jupyter notebook for training the YOLOv11 segmentation model using the provided dataset. Includes proper configuration of hyperparameters, smooth training process without errors, convergence monitoring, and generation of a usable model checkpoint ready for deployment.
- **Hardware Setup (10%):** Basic correct connection and configuration of Raspberry Pi and camera module, ensuring stable video capture and data flow during inference.
- **Software Functionality (20%):** Real-time operation of the segmentation system with accurate detection and overlay of segmentation masks on camera feed. Efficient processing without crashes or freezes. Proper handling of errors or failures during runtime.
- **Project Understanding and Documentation (10%):** Clear explanations of the training steps, model architecture, system workflow, and usage instructions. Ability to interpret model outputs and troubleshoot common issues.

---

13. Languages Available:

---

14. Contributors:

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.